National College of Ireland

Bachelor's in Computing

2023/2024

Gamal Silvio Gullifa

20255195

x20255195@student.ncirl.ie

ElectricVision

Google Colab
https://colab.research.google.com/drive/1jhRxDHSEF7fLG_H_AqQmcuR1objJJqh0?usp=drive_link

ScreenRecording Presentation Meeting with Gamal Gullifa-20240517_233641-Meeting Recording.mp4

Technical Report

Contents

Executive Summary

This report will cover from beginning to end the development of Gamal Gullifa's final computing project about "ElectricVision". After four years studying computing at NCI, I am entitled to develop a program in order to demonstrate acquired skills during my student career.

During the first pages I will introduce you to my ideas in order to predict or more professional "to forecast" the price of the energy consumed in our near future, and probably will answer to some of your questions, what is this project about? Why have I chosen to work on it? How is it related to Computing? To give you a hint on this summary, during my third year my course had a module on Machine Learning, and I got my attention straight away.

Continuing with the report you will find my objectives, collecting data, creating models and different experiments to forecast the value of the consumed energy in the future.

There will be an explanation of the technologies used, as Python (which we have learned during college) and TensorFlow (which I have acquired after third year, when I got my interest in Machine Learning).

I will take you through the system requirements, what is accepted for this project and what is not, how the data has to be collected and what requisites it needs in order to be useful when predicting the price of the energy consumed.

The design and architecture stage gets a bit more technical and I will go deep into the details, explaining deep learning algorithms and approaches as RNNs, CNN 1D, NBEATS and more.

After we have gone through the non-technical and technical details we will be good to go into the implementation and testing stage to evaluate the project, going to the important part, see the program working and predicting the data.

At the end you will see future steps, as this is the mid-point presentation, there will be many more things and finally you can read the conclusion.

1.0 Introduction

1.1. Background

During my third year of my Bachelors, I had nine different modules and one of them was my favourite, “Introduction to Artificial Intelligence and Machine Learning”. During that semester we have learned the foundations of AI and ML, to program AI and ML, and to analyse data (data analytics).

Machine Learning has existed since a long time, but it seems normal to hear about it nowadays frequently. During 2022 and 2023 there was a “Boom” with ML, due to the creation of different self-learning applications as “Chatbots”(Chat GPT), image recognition, product recommendations, etc. All this satisfactory application got my interest and went through reading and studying articles, and assisting to special courses and classes to learn more about them, how they were created, their source code, how the human thinks and how the machine thinks and learn.

Although I have chose this project in September 2023, during this semester I had the module IT Governance, Security & Ethics, where I had to do a final job presentation on Machine Learning ethics, and has inspired me to keep connected with AI & ML.

All these factors have pushed me closer to AI & ML, and made me desire to pursue a Machine Learning Masters when completing this Bachelors. Those are the main reasons why I chose to focus on ML on this project

1.2. Aims

This final project aims will work on the last semester real time data to predict the price of electricity consumption.

Energy prices vary every minute, how much users will pay at the end of the day (or month) changes all the time. There are several factors which modify the price of the consumed energy, if the energy is consumed during daylight, if it is night time, if it is winter or summer, if it is an special day as New Years, Christmas or even the final of the world cup. The price of the energy depends on the society, if their culture have them starting to work earlier or later, and like those we can find thousands of components of why the price changes every minute.

How predicting the price of the consumed energy is related to computing? This project aims to use Machine Learning, and to create ten (10) different models to predict the price of consumed energy. In ML we can find endless models to predict different values however depending on seasonality and specifications of the data, some models are expected to perform better than others.

This final project will work with energy consumption prices but it will really focus on how to build those models, learn how they work, learn why there are different models and there is no a better model overall as they depend on what they want to predict.

Main objectives for Mid-Presentation

- Collect the data: Research through electricity companies, Kaggle, Amazon Web Services Datasets, Google datasets and choose the best data set for our project. On this point, it is important to explain that Machine Learning uses thousands and even millions of values in order to work smoothly. When predicting values, we need as much data as possible to train our models so they will be able to provide the most accurate forecast.
- Writing a pre-processing function to prepare the time series data. Visualising information with Pandas. Visualising time series with Python CSV.
- Creating train and test splits for the time series (I have come up with two possible ways, one of them won't be good for this project, and the other is perfect to predict energy price. Explain why one of the methods is correct and the other is not, even though both are well-known methods used nowadays.
- Create Plotting function to visualise the price of energy consumed in a graphic.
- Start Setting up Multiple Time Series Modelling Experiments.
- Naïve Model, baseline model experiment to start predicting the price of the energy consumption.
- Implementing MASE (Main Absolute Scaled Error) with TensorFlow (and TF introduction).
- Develop functions to evaluate the forecast.
- Formatting data again, in order to use them with windowing and horizon.
- Go into detail in Window and Horizon (Explain how many days of data do we need to train our model in order to predict the future value).

Main objectives for the Final Presentation

- Build ten (10) different Time Series Modelling Experiments
- Replicate the N-Beats algorithm using TensorFlow
- Evaluate the data and information with from the models.
- Compare models and decide which is better for ElectricVision.

1.3. Technology

Google Colab. It is a notebook to write text and code in the same document. It is used to explain concepts, code and execute it at the same time. It does not need to install any software or libraries which makes it lighter and easy to use. It can be shared in Google Drive with different people, so everyone can edit the notebooks at the same time. It is similar to Jupyter Notebook, but in the cloud.

Coding: Python

Libraries:

- Pandas. It is the main library when working in Machine Learning. It is a powerful and fast library to manipulate, analyse and visualise data. Pandas is built on NumPy and matplotlib (ElectricVision will make use of both). NumPy focuses on 2D and 3D arrays, Pandas focuses on dataframe which is an array to store data and manipulate it at a fast speed in rows and columns.
- TensorFlow is an open source framework used to build Machine Learning and Deep Learning Models (I will use it to build the ten model experiments). It helps to train and

execute neural networks, natural language processing and digit classification. It facilitates the estimation of outputs (The forecasting and predictions).

1.4. Structure

The structure of this report includes five (5) sections (Introduction, System, Conclusions, Further Development or Research, References and Appendices).

The introduction provides an overview of the idea, but it is exact when talking about the objectives of ElectricVision. It answers the main questions as what is the topic for this final project (Forecasting of energy prices), what are the main objectives, and what technologies are implemented.

System is the main section of this report, it goes over all the requirements, what is essential to build ElectricVision, the it has the Use Case Diagrams to graphically represent the system including the actors, actions, roles, functions. This system describes the ten (10) models and experiments built in Electric Vision, its design, architecture and implementation and finally the testing of Electric Vision and evaluation of the behaviour.

Conclusions will end the investigation and building of this project, closing out with learned thoughts, considering how the data was collected, formatted and the first model was built.

Further Development and Research will include the next steps of this project. This is only the mid-point presentation, and it only includes the half of the project and 1 experiment, so in this stage will show what is left to be developed and built.

References will include two books that I have used to develop my program and several internet sources which were helpful.

Appendices will include the proposal submitted earlier on this module, where I explained my ideas, and changes carried out.

2.0 System

2.1. Requirements

- Data Collection: Get the data is a requirement for any machine learning project and it is essential for this project. Downloading a CSV file(comma separated values) will help to manipulate the data easily.
- Format the data: Data cannot be simply used as received. Machine Learning works with thousand to millions of information in order to get trained and the estimate or in our case to predict the price of the energy to be consumed in the future. For this reason it is compulsory to have the data in the correct format with no errors. I will explain deeper in the section 2.1.2 Data Requirements.
- Investigation of different model evaluations, there are endless models to predict in Machine Learning, to predict the weather, traffic, time, etc. Every model work differently being better or worse for different projects (It does not mean that they could

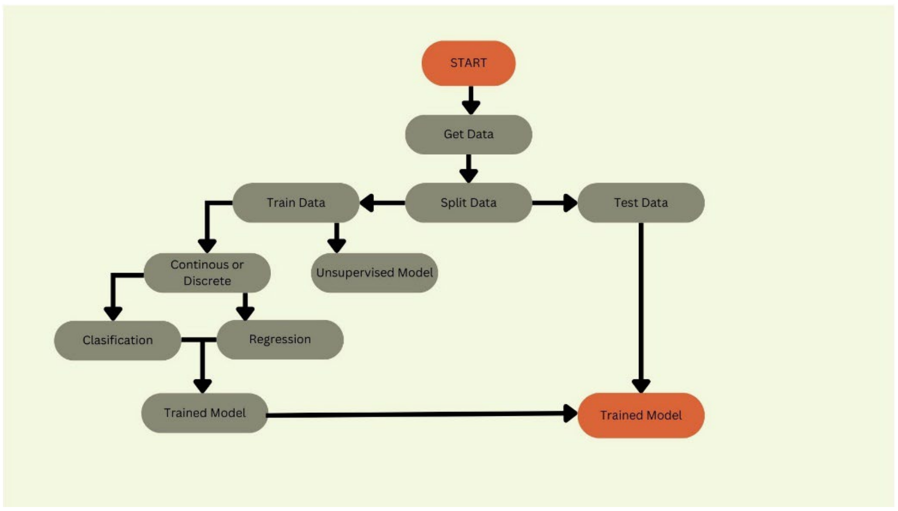
be good or bad models, but it would not be appropriate to use a weather forecast for traffic prediction).

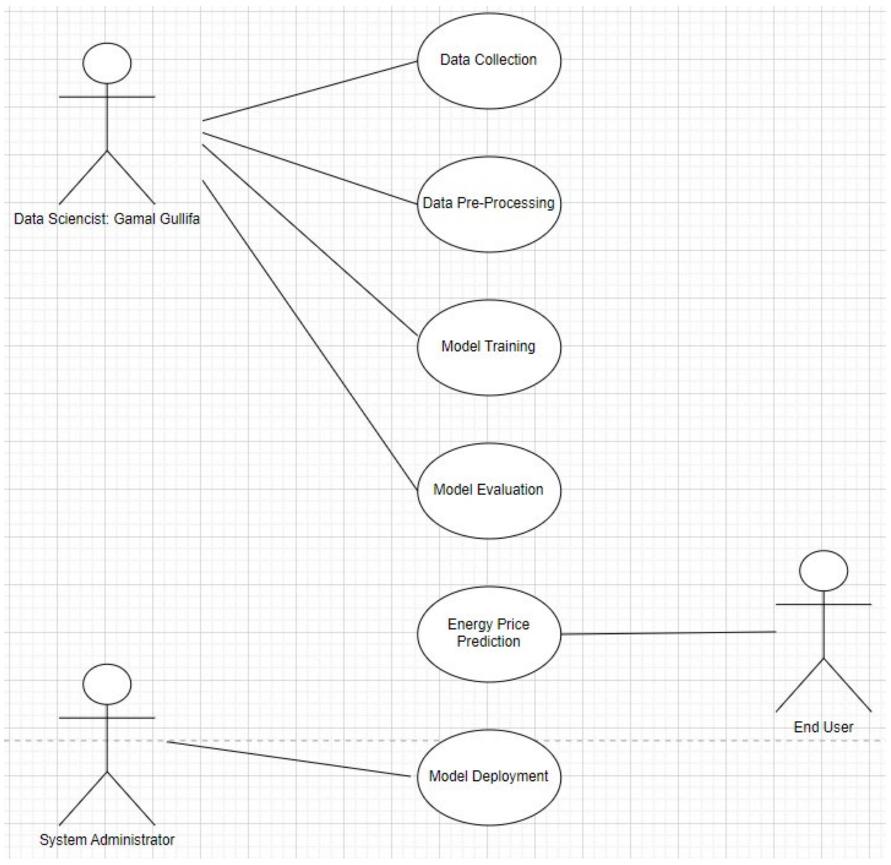
- Model Evaluation: After the creation of the model we evaluate the data obtained with metrics, MAE, MSE, RMSE, we will go over them and again some are easier to use, some are going to provide a better result when evaluating the data.
- ElectricVision should provide prediction in a fast speed, so it will be built with Pandas and TensorFlow, instead of using Python For Loops

2.1.1. Functional Requirements

- For this mid-point presentation, the application must pull the data collected with Pandas and Python's CSV module, encountering no issues.
- Column titles to be renamed to a simple name (Price).
- Show clear graphic of the time series data and predictions with Matplotlib.
- Create a scikit learn model, in order to split the data, having train data and test data.
- Create a function to plot the data in a graphic with the experiments.
- Create Naïve Model, and test it.
- Import and use TensorFlow to evaluate the Naïve Model forecasting.
- Create a function to take the forecasting and return the evaluations.
- Comprehend Windows and Horizons (How many days of the collected data are going to be used in order to provide a prediction).
- Create function to view Numpy arrays as windows

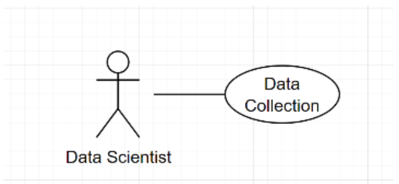
2.1.1.1. Use Case Diagram





2.1.1.2. Requirements, Description and Priority, Use Case (Scope, Diagram, Flow Description: Pre-Condition and Activation, Main Flow, Alternate Flow, Exceptional Flow, Termination, Post Condition).

- UCR1 (User Requirement 1): **Data Collection**:
  DESCRIPTION: To obtain half-hourly data of the prices of the consumed energy during one semester of the last year.
  PRIORITY: High.
  SCOPE: Get all the data needed to be trained and tested afterwards.
  Diagram:



Precondition: Data has to be uploaded in internet, having at least 5000 different values (rows). At least daily values.
Activation: This started during the beginning of the development of the project, when I had to research for the data.

Main Flow:

- ♦ Research for datasets on "Kaggle, UCI ML Repository, AWS, Google's Datasets Search, Microsoft Datasets, energy company's datasets".
- ♦ Review the dataset if the comply with re precondition (>5000 values), and at least daily values (Got half-hourly values and +7000 rows).
- ♦ Merge excel sheets into only one, tidy data chronologically.
- ♦ Transform dataset into CSV File.
- ♦ Upload the data into GitHub in order to make it available for Google Colab.

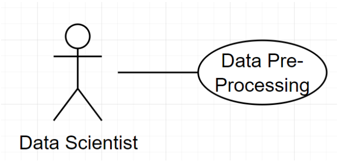
Exceptional Flow:

- ♦ Google Colab found errors because data was not in CSV Format.
- ♦ Dates were mixed.
- ♦ Raw URL link needed from GitHub.
- ♦ Reviewed the errors and fixed the problems.

Termination: !wget command in Google Colab and the system gets the data.

Post Condition: Google Colab is ready to work on the dataset and display it.

- UCR2 (User Requirement 2): **Data Pre-Processing**:
  DESCRIPTION: Data collected was received, now it has to be in perfect conditions to be used. It will be "cleaned" and formatted to be used with Pandas, Python and TensorFlow.
  PRIORITY: High.
  SCOPE: Get ready the data to be used with Python, Pandas and TF. (Example, TF only reads float32).
  Diagram:



Precondition: Data had to be collected. Titles to be formatted to be known. (e.g price inc VAT -> Price). Import Pandas. Import Tensor Flow.
Activation: This started after the collection of the data, when I formatted it into the desired process.

Main Flow:

- ♦ Import Pandas
- ♦ Read and visualise the data
- ♦ Format column titles
- ♦ Import TensorFlow
- ♦ Format values into float32
- ♦ Visualise data

- ♦ Confirm data is ok to be worked on.

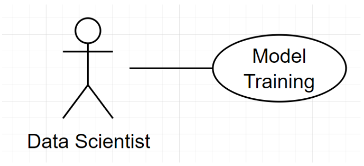
Exceptional Flow:

- ♦ Found titles I did not like as Price_Inc_Vat (too long and "difficult" to type when programming.
- ♦ Formatted the titles.
- ♦ Found issues when tried to visualise float64 values in TF.
- ♦ Formatted values to float32.

Termination: Visualise data and check if data is correct, I do it for the 10 first and last values. Checking column titles as well.

Post Condition: Google Colab has the data ready to be worked on, in correct formats to start creating the first functions.

- <u>UCR3</u> (User Requirement 3): **<u>Model Training</u>**:
  DESCRIPTION: Develop ten (10) different models to predict the price of consumed energy. During this mid-presentation, only the naïve model is completed.
  PRIORITY: High.
  SCOPE: Create trained models to predict the price of energy.
  Diagram:



Precondition: Data had to be collected, formatted and visualised.
Activation: This use case gets activated when the data gets split into train data and test data.

Main Flow:

- ♦ Create sets model.
- ♦ Create train and tests splits (80% and 20%).
- ♦ Create function to plot the time series sets.
- ♦ Create First Model (Naïve Model) following its formula.
- ♦ Offset values of index of 1.

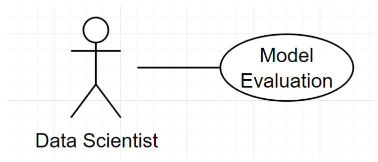
Exceptional Flow:

- ♦ Found that I cannot mix train data and test data for this project as we are predicting for the future.
- ♦ Create functions to have the first 80% of the data to be trained, and the last 20% of the data to be tested.

Termination: Naïve Model is finished and ready to make the first baseline prediction.

Post Condition: We are ready to evaluate the model.

- **UCR4** (User Requirement 4): **Model Evaluation**:
  DESCRIPTION: Define the regressions metrics to evaluate the data, to make sure that the data we collected being used in the functions created are accurate and reliable.
  PRIORITY: High.
  SCOPE: Implementation of MASE to evaluate data.
  Diagram:



Precondition: Data collected, formatted, visualised, model functions created.
Activation: This use case gets active when the MASE function is called in order to evaluate the data.

Main Flow:

- ♦ Chose the best regression metrics for the naïve model (MAE, MSE, RMSE, MAPE, MASE)
- ♦ Chose Mase.
- ♦ Develop the function to be implemented in TF.
- ♦ Analyse the scaled error
- ♦ Run function and check if it worked.
- ♦ Saw low results which is good.

Termination: Google Colab displays the evaluation by MASE.

Post Condition: Ready for the first energy prediction value.

2.1.2. Data Requirements

2.1.3. User Requirements

2.1.4. Environmental Requirements

2.1.5. Usability Requirements

2.2. Design & Architecture

TEN FORECASTING MODELS EXPERIMENTS (Analysing which model performs the best):

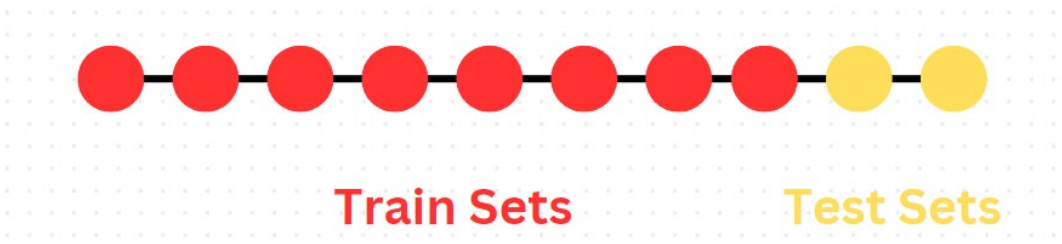
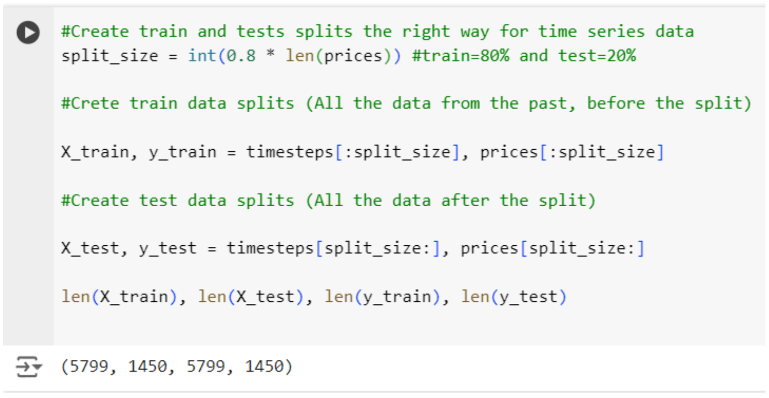


Figure 1 Train and Test Sets



```
#Create train and tests splits the right way for time series data
split_size = int(0.8 * len(prices)) #train=80% and test=20%

#Crete train data splits (All the data from the past, before the split)

X_train, y_train = timesteps[:split_size], prices[:split_size]

#Create test data splits (All the data after the split)

X_test, y_test = timesteps[split_size:], prices[split_size:]

len(X_train), len(X_test), len(y_train), len(y_test)
```

```
(5799, 1450, 5799, 1450)
```

The experiments will start simple, and the complexity will be increased in every experiment.

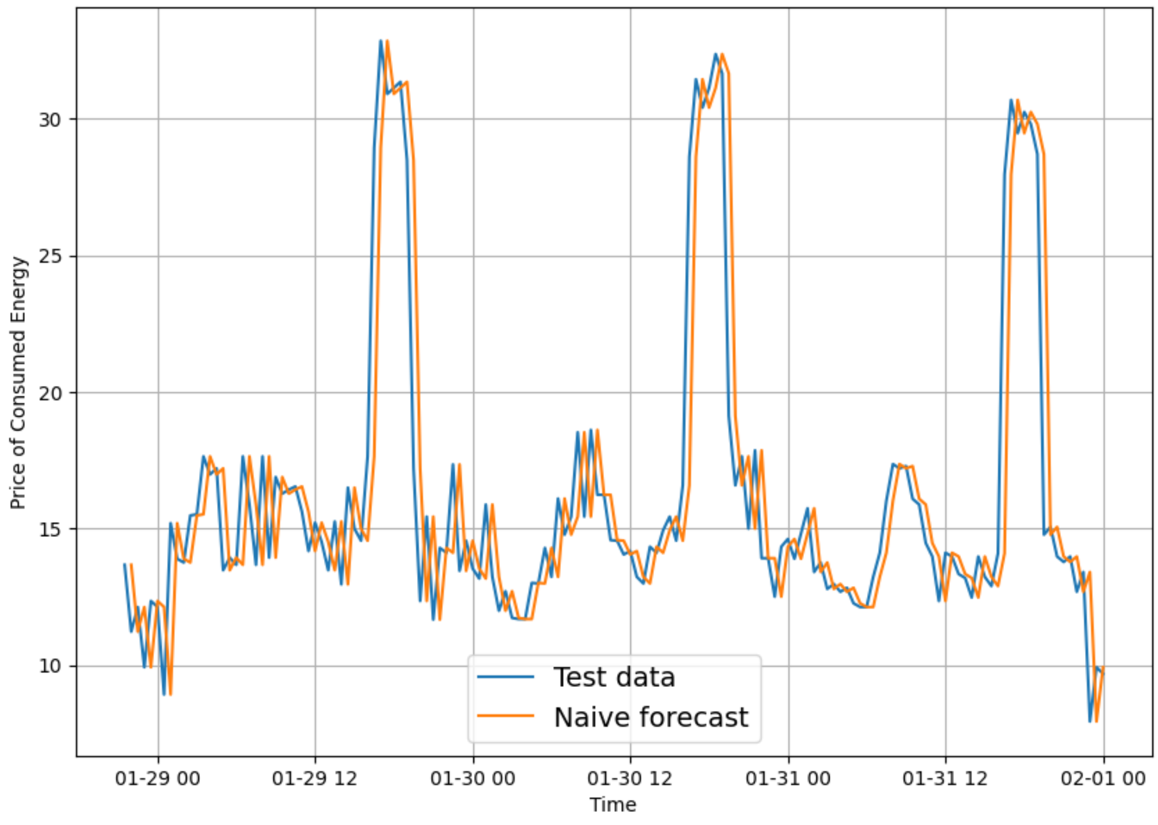
<u>Experiment 0 “Naïve Model”:</u> It is 0 because it is a baseline, it will help to develop the future experiments.

“For Naïve forecast, we simply set all forecast to be the value of the last observation”.

It blindly predicts the previous time step as the next time step. The naïve model formula is:

$$\hat{y}_t = y_{t-1}$$

The prediction at timestep t(y-hat) is equal to the value at timestep t-1 (previous timestep) – this is for horizon of 1.



For our horizon of one, the naive forecast is predicting the previous time step as the next steps value.

The Naive forecast appears to align closely with the test data, exhibiting a pattern that mirrors the observed trends. This alignment is expected, given that our model does not incorporate any adjustments or shifts to the test data. The recorded energy prices exhibit a variation of one index. Notably, the orange line consistently trails behind the blue line, representing the actual data. This discrepancy highlights the predictive nature of the naive model.

Experiment 1 Dense Model (Horizon = 1, Window = 7)

Experiment 2 Dense Model 2.0 (Different Window Size (Horizon = 1, Window = 30))

Experiment 3 Dense Model 3.0 (Different Window Size (Horizon = 7, Window = 30))

Experiment 4 Conv1D Sequence Problems

Experiment 5 LSTM Sequence problems

Experiment 6 Dense Model 3.0 (Multivariate Data)

Experiment 7 N-BEATS Algorithms

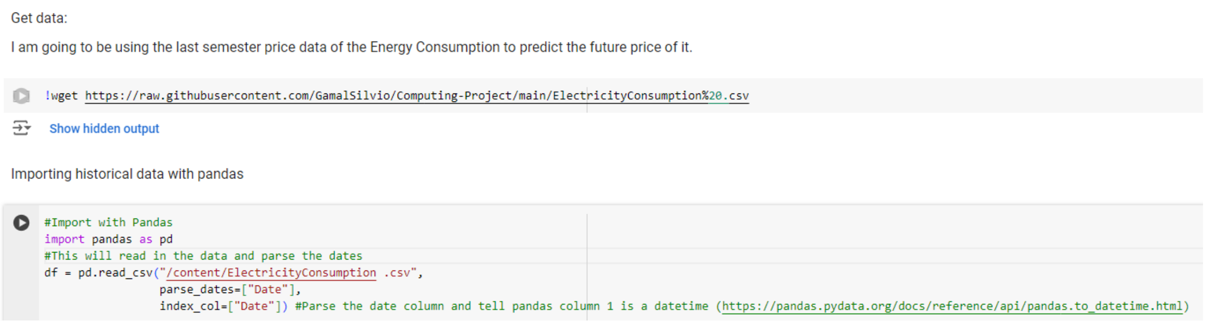
Experiment 8 Ensemble (Multiple Models Stacked Together)

Experiment 9 Future Predict Model

Experiment 10 Dense Model 4.0

2.3. Implementation and Graphical User Interphase GUI

Getting data, importing Pandas library and reading the data from Pandas.



Get data:

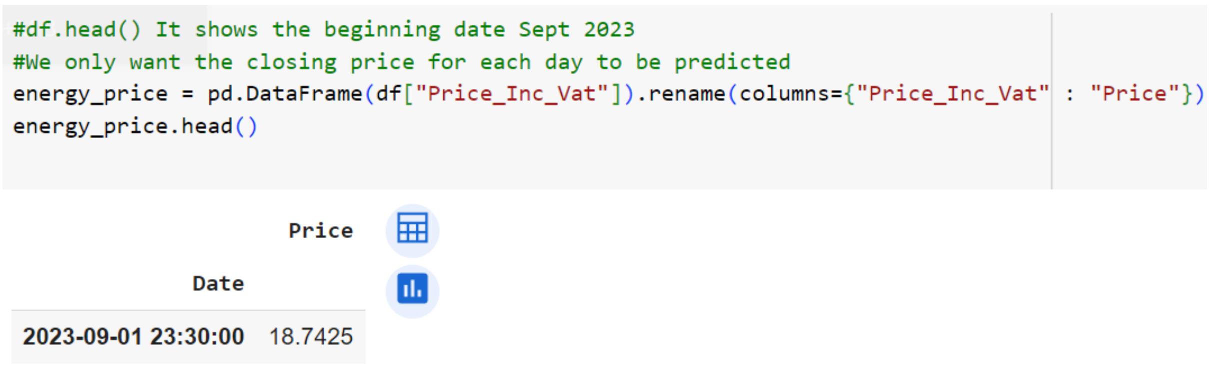
I am going to be using the last semester price data of the Energy Consumption to predict the future price of it.

```
!wget https://raw.githubusercontent.com/GamalSilvio/Computing-Project/main/ElectricityConsumption%20.csv
```

Importing historical data with pandas

```
#Import with Pandas
import pandas as pd
#This will read in the data and parse the dates
df = pd.read_csv("/content/ElectricityConsumption .csv",
                 parse_dates=["Date"],
                 index_col=["Date"]) #Parse the date column and tell pandas column 1 is a datetime (https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html)
```

Formatting the columns titles (Price_Inc_Vat to Price), in order to have a clearer program and easier to typer.

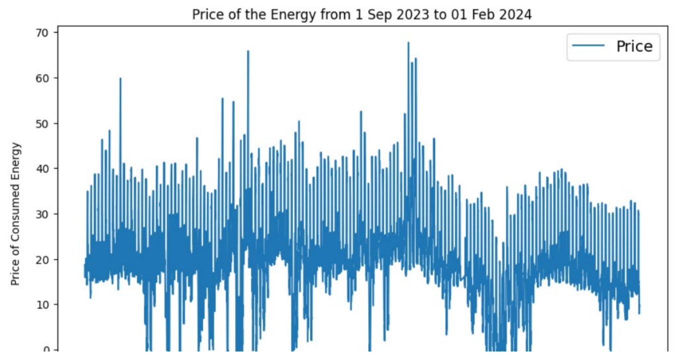


```
#df.head() It shows the beginning date Sept 2023
#We only want the closing price for each day to be predicted
energy_price = pd.DataFrame(df["Price_Inc_Vat"]).rename(columns={"Price_Inc_Vat" : "Price"})
energy_price.head()
```

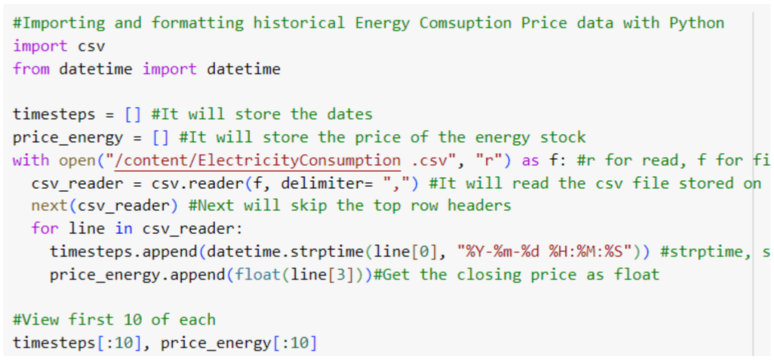
	Price
Date	
2023-09-01 23:30:00	18.7425

Importing Matplotlib and plotting the data of the energy consumed from Sept 2023 to Feb 2024 into a graphic.

```
import matplotlib.pyplot as plt
energy_price.plot(figsize=(10,7))
plt.ylabel("Price of Consumed Energy")
plt.title("Price of the Energy from 1 Sep 2023 to 01 Feb 2024")
plt.legend(fontsize=14);
```

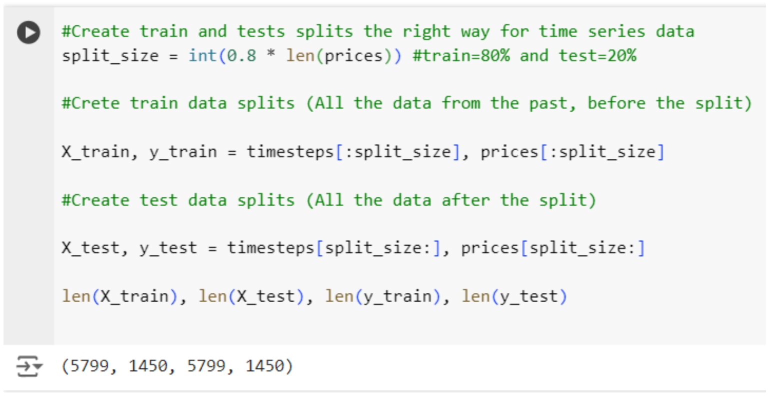


Importing time series again but this time from Python CSV.



```
#Importing and formatting historical Energy Comsuption Price data with Python
import csv
from datetime import datetime

timesteps = [] #It will store the dates
price_energy = [] #It will store the price of the energy stock
with open("/content/ElectricityConsumption .csv", "r") as f: #r for read, f for fi
  csv_reader = csv.reader(f, delimiter= ",") #It will read the csv file stored on
  next(csv_reader) #Next will skip the top row headers
  for line in csv_reader:
    timesteps.append(datetime.strptime(line[0], "%Y-%m-%d %H:%M:%S")) #strptime, s
    price_energy.append(float(line[3]))#Get the closing price as float

#View first 10 of each
timesteps[:10], price_energy[:10]
```

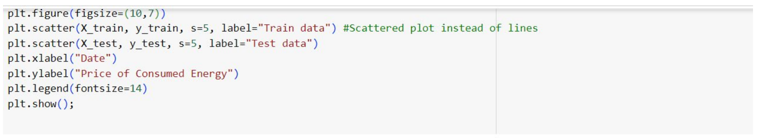
Creates the train and test sets for the times series, as ML uses data between thousands and millions, it is better to have a higher train set with 80% and only 20% for the tests.



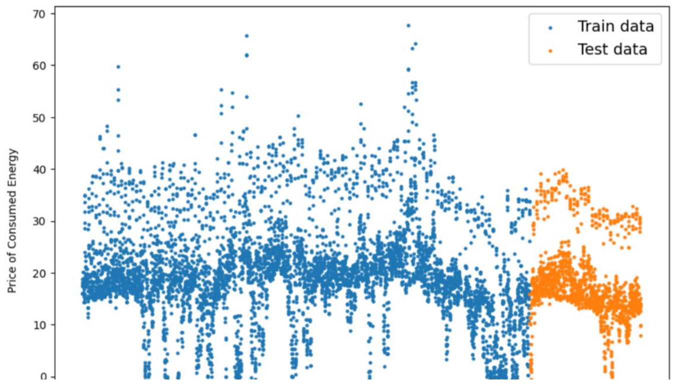
```
#Create train and tests splits the right way for time series data
split_size = int(0.8 * len(prices)) #train=80% and test=20%

#Crete train data splits (All the data from the past, before the split)

X_train, y_train = timesteps[:split_size], prices[:split_size]

#Create test data splits (All the data after the split)

X_test, y_test = timesteps[split_size:], prices[split_size:]

len(X_train), len(X_test), len(y_train), len(y_test)
```

```
(5799, 1450, 5799, 1450)
```

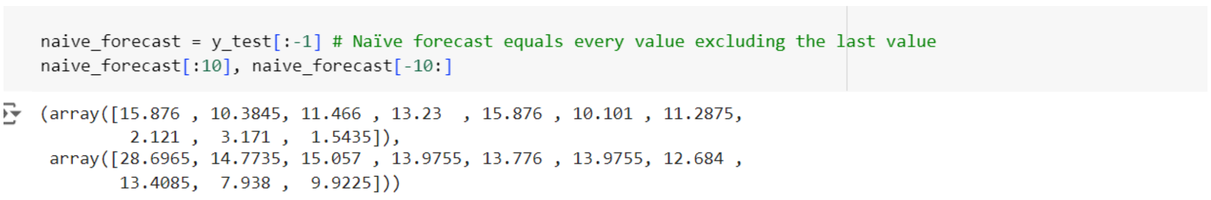
Plotting the data into 80% train sets and 20% test sets, keeping the first period as train, and the last months as tests.



```
plt.figure(figsize=(10,7))
plt.scatter(X_train, y_train, s=5, label="Train data") #Scattered plot instead of lines
plt.scatter(X_test, y_test, s=5, label="Test data")
plt.xlabel("Date")
plt.ylabel("Price of Consumed Energy")
plt.legend(fontsize=14)
plt.show();
```



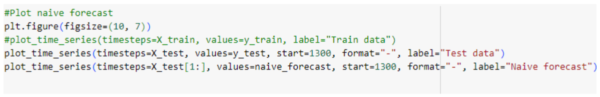
Naïve Model and formula which rests 1:



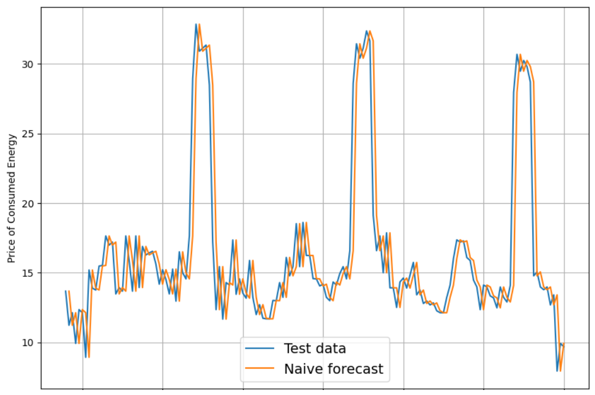
```
naive_forecast = y_test[:-1] # Naïve forecast equals every value excluding the last value
naive_forecast[:10], naive_forecast[-10:]
```

```
(array([15.876 , 10.3845, 11.466 , 13.23   , 15.876 , 10.101 , 11.2875,
         2.121 ,  3.171 ,  1.5435]),
 array([28.6965, 14.7735, 15.057 , 13.9755, 13.776 , 13.9755, 12.684 ,
        13.4085,  7.938 ,  9.9225]))
```

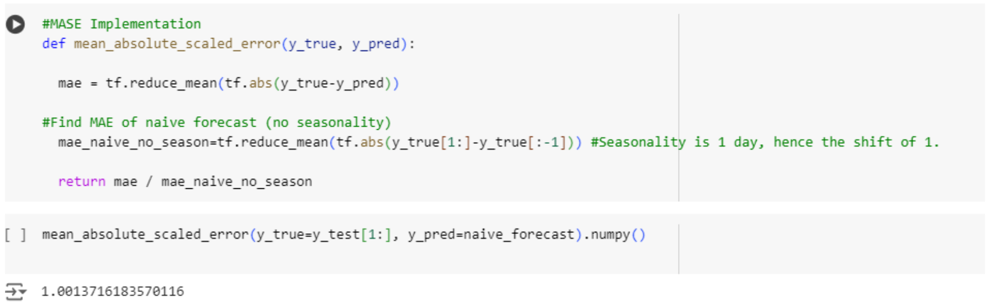
Plot Naïve formula



```
#Plot naive forecast
plt.figure(figsize=(10, 7))
#plot_time_series(timesteps=X_train, values=y_train, label="Train data")
plot_time_series(timesteps=X_test, values=y_test, start=1300, format="-", label="Test data")
plot_time_series(timesteps=X_test[1:], values=naive_forecast, start=1300, format="-", label="Naive forecast")
```



MASE Implementation to evaluate the data from Naïve Model

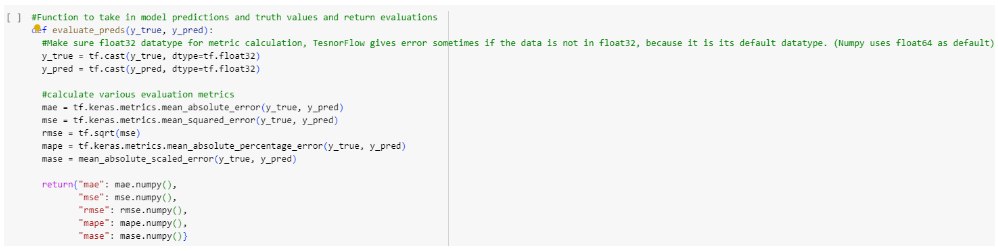


```
#MASE Implementation
def mean_absolute_scaled_error(y_true, y_pred):

  mae = tf.reduce_mean(tf.abs(y_true-y_pred))

#Find MAE of naive forecast (no seasonality)
  mae_naive_no_season=tf.reduce_mean(tf.abs(y_true[1:]-y_true[:-1])) #Seasonality is 1 day, hence the shift of 1.

  return mae / mae_naive_no_season
```

```
[ ] mean_absolute_scaled_error(y_true=y_test[1:], y_pred=naive_forecast).numpy()
```

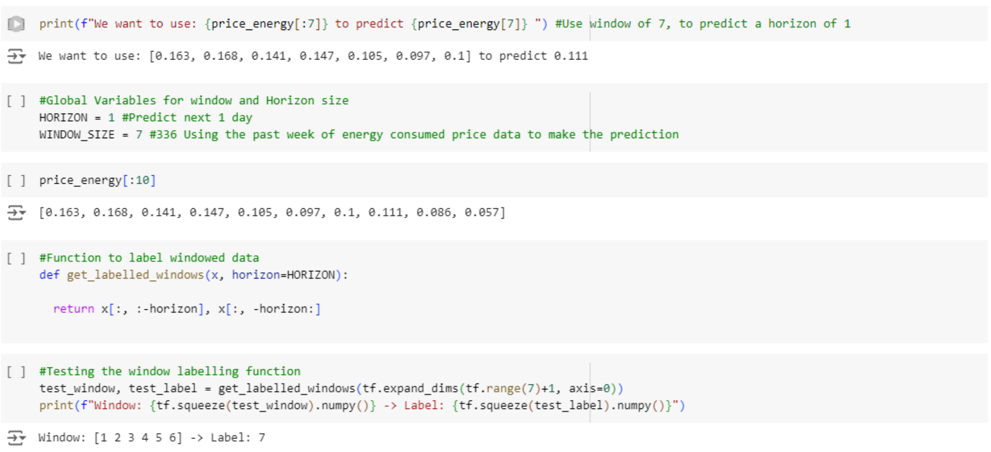
```
1.0013716183570116
```

Function to take in model predictions and truth values and return evaluations with TensorFlow



```
[ ] #Function to take in model predictions and truth values and return evaluations
    def evaluate_preds(y_true, y_pred):
      #Make sure float32 datatype for metric calculation, TesnorFlow gives error sometimes if the data is not in float32, because it is its default datatype. (Numpy uses float64 as default)
      y_true = tf.cast(y_true, dtype=tf.float32)
      y_pred = tf.cast(y_pred, dtype=tf.float32)

      #calculate various evaluation metrics
      mae = tf.keras.metrics.mean_absolute_error(y_true, y_pred)
      mse = tf.keras.metrics.mean_squared_error(y_true, y_pred)
      rmse = tf.sqrt(mse)
      mape = tf.keras.metrics.mean_absolute_percentage_error(y_true, y_pred)
      mase = mean_absolute_scaled_error(y_true, y_pred)

      return{"mae": mae.numpy(),
             "mse": mse.numpy(),
             "rmse": rmse.numpy(),
             "mape": mape.numpy(),
             "mase": mase.numpy()}
```

Windows and Horizons:



```
print(f"We want to use: {price_energy[:7]} to predict {price_energy[7]} ") #Use window of 7, to predict a horizon of 1
```

```
We want to use: [0.163, 0.168, 0.141, 0.147, 0.105, 0.097, 0.1] to predict 0.111
```

```
[ ] #Global Variables for window and Horizon size
    HORIZON = 1 #Predict next 1 day
    WINDOW_SIZE = 7 #336 Using the past week of energy consumed price data to make the prediction
```

```
[ ] price_energy[:10]
```

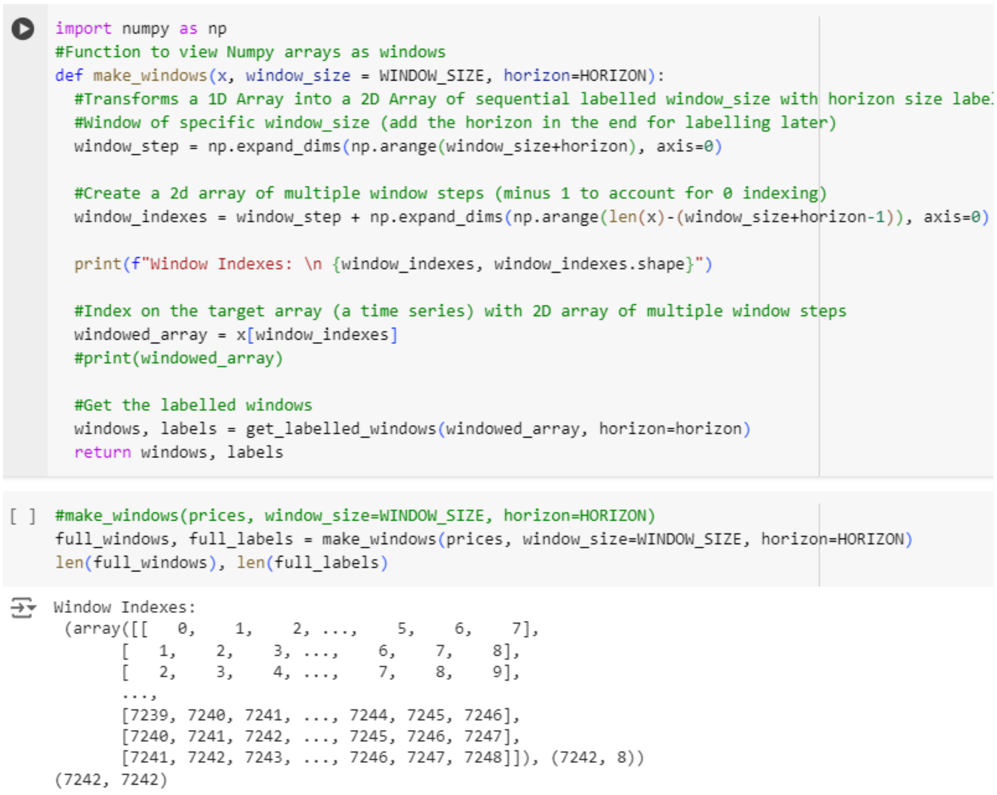
```
[0.163, 0.168, 0.141, 0.147, 0.105, 0.097, 0.1, 0.111, 0.086, 0.057]
```

```
[ ] #Function to label windowed data
    def get_labelled_windows(x, horizon=HORIZON):

      return x[:, :-horizon], x[:, -horizon:]
```

```
[ ] #Testing the window labelling function
    test_window, test_label = get_labelled_windows(tf.expand_dims(tf.range(7)+1, axis=0))
    print(f"Window: {tf.squeeze(test_window).numpy()} -> Label: {tf.squeeze(test_label).numpy()}")
```

```
Window: [1 2 3 4 5 6] -> Label: 7
```

Transforming the 1D arrays to 2D arrays of sequential labelled window_size horizon size labels.

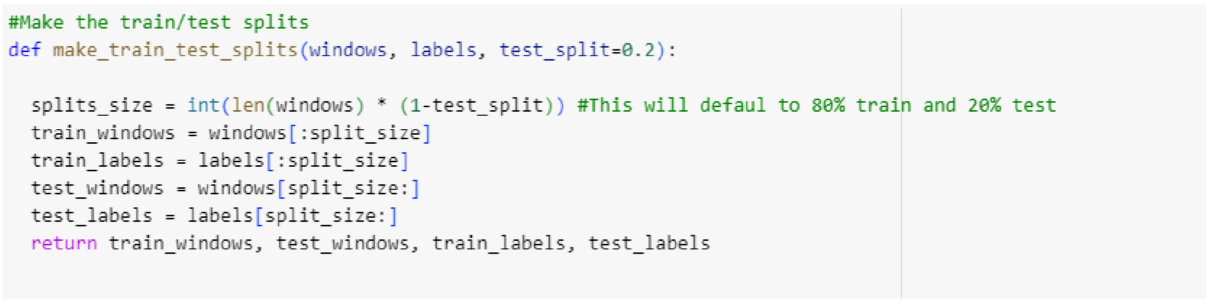


```
import numpy as np
#Function to view Numpy arrays as windows
def make_windows(x, window_size = WINDOW_SIZE, horizon=HORIZON):
  #Transforms a 1D Array into a 2D Array of sequential labelled window_size with horizon size labe
  #Window of specific window_size (add the horizon in the end for labelling later)
  window_step = np.expand_dims(np.arange(window_size+horizon), axis=0)

  #Create a 2d array of multiple window steps (minus 1 to account for 0 indexing)
  window_indexes = window_step + np.expand_dims(np.arange(len(x)-(window_size+horizon-1)), axis=0)

  print(f"Window Indexes: \n {window_indexes, window_indexes.shape}")

  #Index on the target array (a time series) with 2D array of multiple window steps
  windowed_array = x[window_indexes]
  #print(windowed_array)

  #Get the labelled windows
  windows, labels = get_labelled_windows(windowed_array, horizon=horizon)
  return windows, labels
```

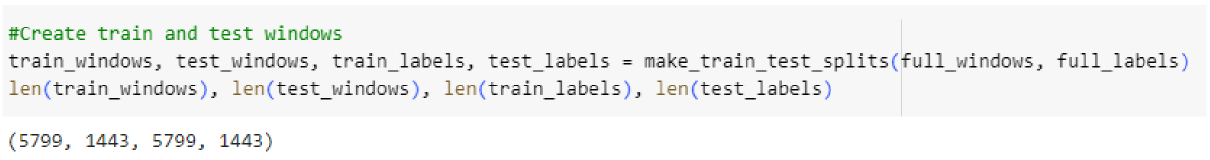
```
[ ] #make_windows(prices, window_size=WINDOW_SIZE, horizon=HORIZON)
    full_windows, full_labels = make_windows(prices, window_size=WINDOW_SIZE, horizon=HORIZON)
    len(full_windows), len(full_labels)
```

```
Window Indexes:
 (array([[   0,    1,    2, ...,    5,    6,    7],
        [   1,    2,    3, ...,    6,    7,    8],
        [   2,    3,    4, ...,    7,    8,    9],
        ...,
        [7239, 7240, 7241, ..., 7244, 7245, 7246],
        [7240, 7241, 7242, ..., 7245, 7246, 7247],
        [7241, 7242, 7243, ..., 7246, 7247, 7248]]), (7242, 8))
(7242, 7242)
```

Turning Windows into training and test splits. Split matching pairs of windows and labels into train and tests splits.



```
#Make the train/test splits
def make_train_test_splits(windows, labels, test_split=0.2):

  splits_size = int(len(windows) * (1-test_split)) #This will defaul to 80% train and 20% test
  train_windows = windows[:split_size]
  train_labels = labels[:split_size]
  test_windows = windows[split_size:]
  test_labels = labels[split_size:]
  return train_windows, test_windows, train_labels, test_labels
```



```
#Create train and test windows
train_windows, test_windows, train_labels, test_labels = make_train_test_splits(full_windows, full_labels)
len(train_windows), len(test_windows), len(train_labels), len(test_labels)
```

```
(5799, 1443, 5799, 1443)
```

.

2.4. Testing

I manage the functions and run one after the other and proceeded with manual testing. Had to do several changes during the development of this applications, however a better testing procedure will be carried out for the next submission.

3.0 Conclusions

As we have reached the end of this mid-point presentation, I was able to collect the data, to build all the functions to be used for the 10 models, and already started to work with the first naïve model.

Data reviewing, code reviewing, paying attention to details, and deep reading is what I take from the non-technical details, a final project is complex to carry out.

Being more technical, the price of the energy consumed as per the naïve model (base line model) shows that the price is the same as the last value received. This model learns from the past values replicating its last one into the future.

I am very excited now to discover how more precise models work forecasting the energy consumption price into the future.

4.0 Further Development or Research

I am just half way from finalizing this project, I have to be working on the other experiments which get more complex.

5.0 References

sklearn.model_selection.train_test_split. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Forecasting: Principles and practices: Chapter 5.8 Evaluating Point Forecast Accuracy. (n.d.). https://otexts.com/fpp3/accuracy.html

MeanAbsoluteScaledError — sktime documentation. (n.d.). https://www.sktime.net/en/stable/api_reference/auto_generated/sktime.performance_metrics.forecasting.MeanAbsoluteScaledError.html

CamDavidsonPilon. (n.d.). *Python-Numerics/TimeSeries/MASE.py at master* https://github.com/CamDavidsonPilon/Python-Numerics/blob/master/TimeSeries/MASE.py

MeanAbsoluteError. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/metrics/MeanAbsoluteError

6.0 Appendices

6.1. Project Proposal

Contents

7.0 Objective

This final project will work with historical and real-time data to predict the national electricity consumption.

The primary objectives include:

- Building prediction models to visualise the usage and demand of electricity.
- Explore temporary problems, understanding when there is more demand for electricity usage and when there are usage anomalies.
- Supervised Machine Learning, this project will count on supervised learning, having data with associated etiquette (how much electricity was used in the past, and how much electricity was used today) to train the models to predict how much electricity will be used in the future.
- Model experiments with Deep Learning, develop and compare different types of neuronal network models to predict the electricity demand. Implementation of different models such as LSTM, CNN 1D, Dense, and algorithm N-BEATS with TensorFlow.
- Obtain historical electricity data, importing it into Pandas and manually with Python, clear the data to work on relevant information needed to calculate the usage of electricity and remove unnecessary data.
- Visualise on a graphic the data to study the behaviour of the demands.
- Work with different types of time series to understand the demand for electricity with only one variable (demand of electricity, and multiple variables such as seasons and events).

This project focuses on the prediction and visualisation of electricity demand utilising deep learning models and historical data.

8.0 Background

As I approach the end of my Bachelor's in Computing, I am considering the , possibility of pursuing a Master's Degree in Artificial Intelligence and Machine Learning offered by our college. After the curriculum's program, I was fascinated by the potential of Artificial Intelligence, Machine Learning and Deep Learning, especially on how they work and process the information.

Electricity is the cornerstone of modern society, allowing almost everything in our daily lives. Predicting the electricity demand for the following month or year could help significantly optimise resource allocation, enhance energy efficiency or even help with grid stability and load balancing.

Consequently, I have decided to dedicate my final project to the prediction of electricity demand. This project not only aligns with the urgent need for predictions in critical sectors, but it also works as an ideal application of AI and machine learning techniques. Through this project, my main objective is to contribute to the advancement of predictive models that can have an impact in the real world.

9.0 State of the Art

During my research, I have found similar applications that focus on prediction utilising Artificial Intelligence and Deep Learning, some examples include the prediction of sales for important brands, weather forecasts, and stock price predictions.

What distinguishes my project is its focus on the combination of supervised learning and deep learning techniques. (As LSTM, CNN 1D, Dense and N-BEATS) to predict the electricity demand. It also stands out because of its emphasis on the detailed analysis of different time series and effective visualisation of the patterns of demands.

The main difference between my idea and other applications lies in the implementation of deep learning models, combined with the process of specific data to predict the electricity demand. This combination with the detailed analysis of multiple temporary variables, sets a great difference from other similar existent projects.

10.0 Technical Approach

Approach to development: This project will adopt an iterative and incremental methodology, based on agile methodology so it can adopt continuous changes and deliver functionalities progressively.

Technology and Tools: Utilize a combination of Python, TensorFlow and Pandas to process the data, models, and visualisation.

Communication with Supervisor: Effective communication and collaboration will be fundamental, to keep a constant flow of information.

Requirements Identification:

Investigation & Analysis: Carry out an exhaustive investigation on different sources, studies, and documentation of information available which contains historical data on electrical usage.

Data Evaluation: Rigorous evaluation of available data sources to determine their quality, relevance, and reliability.

Ethical and Lawful aspects: Strict adherence to the law and how I will manage and process the sensible data related to electricity consumption.

Requirements Breakdown:

Detailed Planification: The detailed working structure, will contain from the data acquisition source until the model validation and results presentation.

Specific Milestones: Identification of crucial milestones, such as the sanitisation of data, model implementation and the interactive visualisation generation.

Project Execution:

Data preparation and cleaning: Importing historical electrical consumption data and carrying out an exhaustive sanitising to guarantee the integrity and utility for the models.

Visualization and interpretation: Clear creation and visualisation, understandable to analyse the electricity demand's patterns and to facilitate the interpretation of the results.

Evaluation and continuous improvement: Constant evaluation seeks the models to be precise, optimising and making changes continuously to improve the predictions.

Documentation and reports:

Record keeping and Monitoring: Detailed progress record, highlighting key decisions and challenges overcome during the development of the project.

Final Report: Elaborate a final and complete report which will present the results, used methodologies, limitations, and recommendations for future investigations.

11.0 Technical Details

Implementation Language and Principal Libraries:

The main programming language for this project will be Python, due to its versatility and wide support for Artificial Intelligence and deep learning. Regarding to the libraries, TensorFlow will be fundamental, one of the most popular tools to build and train Neuronal Network Models. TensorFlow offers diverse functionalities to conduct specific tasks in the Deep Learning area.

In addition to TensorFlow, other libraries such as Pandas for data manipulation, Matplotlib for visualisation, and Scikit-learn for some processes or additional techniques in Machine Learning that will be used for specific time or historical data.

Algorithms and approaches considered for this work:

- **Recurrent Neural Networks (RNNs):** These networks are particularly useful for modelling temporal sequences, as they can retain and use the information from previous status. Variants like Long-Short-Term-Memory (LSTM) or Gated Recurrent Unit (GRU) are commonly chosen due to their capability to capture long-term dependencies in time series data.
- **1D Convolutional Neuronal Networks (CNN 1D):** Though it is common to see this algorithm associated with images, CNN 1D can also be applied to Time Series successfully. It can capture local patterns and important characteristics in sequential data.
- **Attention Mechanisms:** These models allow the network to focus on specific parts of the time series, which can be used to provide a precise forecast, giving major importance to certain relevant periods.
- **Transformers for Time Series**

  The primary focus is to experiment with these architectures, tuning hyperparameters, and trying different layer configurations and optimizers to achieve a model that can effectively predict the behaviour of time series. In addition, it can also explore multiple models or ensemble techniques to enhance better prediction precision.

This project will centre on the comparison of these approaches, evaluating specific data sets performance and selecting the best strategies or even combining strategies to predict precisely in the Electricity consumption context.

12.0 Special Resources Required

Special resources will be required for this project to improve the development and evaluation of models to predict electricity usage.

High-Performance Computing: Time Series predictions, especially when talking about deep learning models, demand significant computational power.

Large Time Series Datasets: A diverse historical dataset is crucial to train and validate models effectively, having access to extensive datasets related to electricity, energy or other relevant topics related to electricity would be good to build solid predictions.

Expertise in Time Series Analysis: Access to domain experts or specialised sources in Time Series analysis, who can provide valuable information or ideas, and guidance for feature selection, model validation, and interpreting model outputs.

13.0 Project Plan

Phase 1: Data Exploration and Understanding

- **Start date:** December 20, 2023
- **End date:** March 3, 2024
- **Details and deliverables:**
    - Download the historical electricity dataset and ensure the quality of information.
    - Initial inspection and data sanitation, Padas utilisation and graphics to identify seasonal patterns, trends, and potential anomalies in electricity consumption.
    - Detailed visualisation of the information collected, to have a better understanding of the electricity demand.

Phase 2: Data Preparation for Modelling

- **Start date:** March 4, 2024
- **End date:** April 27, 2024
- **Details and deliverables:**
    - Data processing functions development, to convert Time Series into the windows and labels for training.
    - Data segmentation into training sets and trials respectively to the temporary sequence.
    - Functions development to evaluate specific metrics for Time Series.
    - Multivariable preparation to enrich available information for modelling.

Phase 3: Data Preparation for Modelling

- **Start date:** April 28, 2024

- **End date:** July 31, 2024
- **Details and deliverables:**
    - Implementation and evaluation of multiple Deep Learning models (LSTM, CNN 1D, N-BEATS), using TensorFlow).
    - Hyperparameters and cross-validation to optimise model performance.
    - Exhaustive model comparison and detailed analysis of the obtained results.
    - Document the methodology implemented for every model and its performance with the electricity dataset.

Phase 4: Model Optimisation and Ensemble Integration

- **Start date:** August 1, 2024
- **End date:** August 4, 2024
- **Details and deliverables:**
    - Construction of ensemble models which combine the best individual models to improve the prediction precision.
    - Intervals prediction implementation to quantify uncertainty in forecasts.
    - Comparative Analysis of ensemble models focusing on improvement of the precision and prediction stability.
    - Preparation of detailed documentation to describe the construction and performance of the ensembled models.
    - Elaborate documentation including complete methodology, final results, limitations and possible areas to improve.
    - Video presentation creation and slides to effectively communicate the key aspects of the project.
    - Exhaustive review of every single step, and findings of the project to refine documentation for the final stage.

Phase 5: Viva Examination (If required)

- **Start date:** August 5, 2024
- **End date:** August 15, 2024
- **Details and deliverables:**
    - Preparation, and in-depth understanding of the project.
    - Be ready to answer questions.
    - Anticipate questions.

14.0 Testing

Technical Testing:

- Validation with Real Data: The prediction models will undergo rigorous testing with historical and real-time electricity consumption data to evaluate the precision and predictive capability.

- Integration Testing: Evaluation of the system´s capacity to integrate multiple predictive models and check if it is working as intended.
- Stability Evaluation: Stability and system robusticity verification when facing variations and electricity demand peaks.

14.1. Ethics Approval Application (only if required)

Not required.